

Réalité augmentée et traitement d'images

Christophe Vestri

Le mardi 28 janvier 2020

Objectifs du cours

- Connaitre/approfondir la RA
- Avoir quelques bases théoriques
- Expérimenter quelques méthodes et outils
- Réaliser un projet en RA
- Evaluation:
 - Présence (20%)
 - Participation en classe (40%)
 - Projet (40%)

Plan du cours

- 7 janvier : Réalité augmentée intro, Unity/Vuforia et projet
 - 15 janvier: Construction application RA, Unity StarWars
 - 28 janvier: Vision par ordinateur (1) et Unity Roll-a-ball
 - 4 février: Vision par ordinateur (2) et Geoloc + projet
 - 11 février?? (-> 3 mars) : QRCode, Résumé et présentation des Projets
-
- **Suite: Cours Cartographie/JS/AR/VR**

Plan Cours 3

- Vision par Ordinateur
 - Introduction
 - Comprendre l'image
 - Modélisation
 - Utilisation
- OpenCV
- Exercice Unity Roll a ball et aide projet

Qu'est-ce que la Vision par Ordinateur

- Wikipedia:

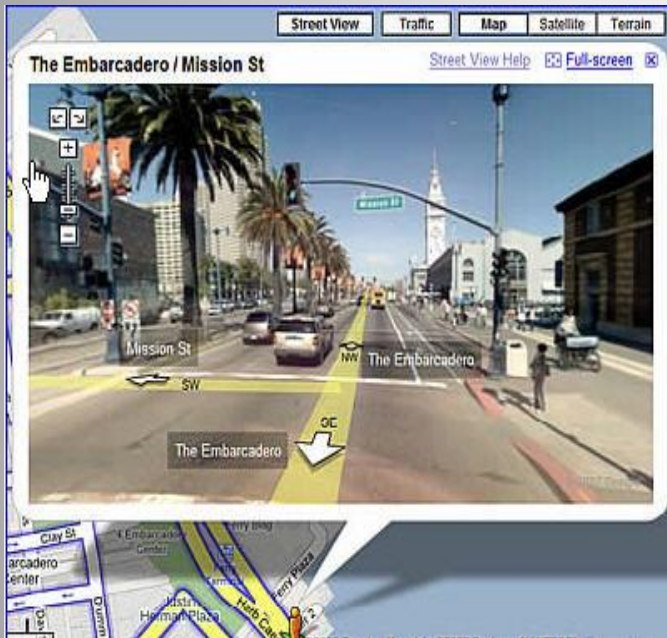
La **vision par ordinateur** (aussi appelée **vision artificielle** ou **vision numérique**) est une branche de l'[intelligence artificielle](#) dont le principal but est de permettre à une machine d'analyser, traiter et comprendre une ou plusieurs images prises par un système d'acquisition (par exemple: [caméras](#), etc.)¹.

- Ma définition:

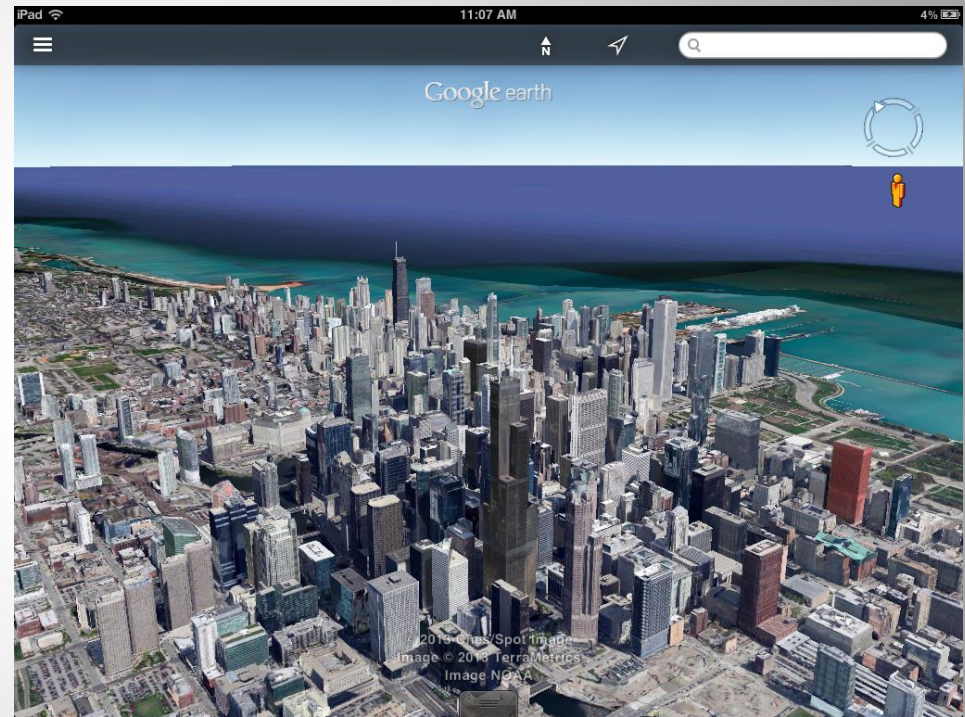
La **vision par ordinateur** regroupe les sciences et techniques permettant aux ordinateurs de **percevoir, voir** et **comprendre** l'environnement capté.

Google

StreetView



Google Earth



Google

Moteur de recherche d'image, classification automatique (deep learning)

Google search results for "chat bleu" (blue cat).

The search bar shows "chat bleu" and the "Images" tab is selected. The results are categorized into four groups:

- Chaton Bleu**: Three small images of blue kittens.
- Chat Bleu Dessin**: Five cartoon drawings of blue cats.
- Chat Gris Bleu**: Three images of blue cats with grey markings.
- Chat Bleu Mar**: One image of a blue cat with wings.

Below these categories, a grid of 15 images shows various blue cats in different poses and settings. The URL at the bottom is: <https://www.google.fr/search?biw=1138&bih=558&tbm=isch&q=chaton+bleu&revid=1592118796&sa=X&ved=0ahUKEwjB5-6jrovMAhUGKQ8KHV3dCFUQ1QIIIHA>

Google

Reconnaissance faciale



Google Glass



Google goggles



Reconstruction 3D (Tango)



Google Car



Autres exemples

- Overview: <http://www.rsipvision.com/>
- Robotique ([liste entreprises](#)):
 - <https://www.aldebaran.com/fr>
 - <http://www.robosoft.com/>
 - Robotique industrielle: [Huget](#)
- Automobile: [Daimler](#), [Mobileye](#)
- Video: <http://360designs.io/>
 - <http://www.video-stitch.com/>
 - [ABlive3D](#), [Tagger](#)

Autres exemples

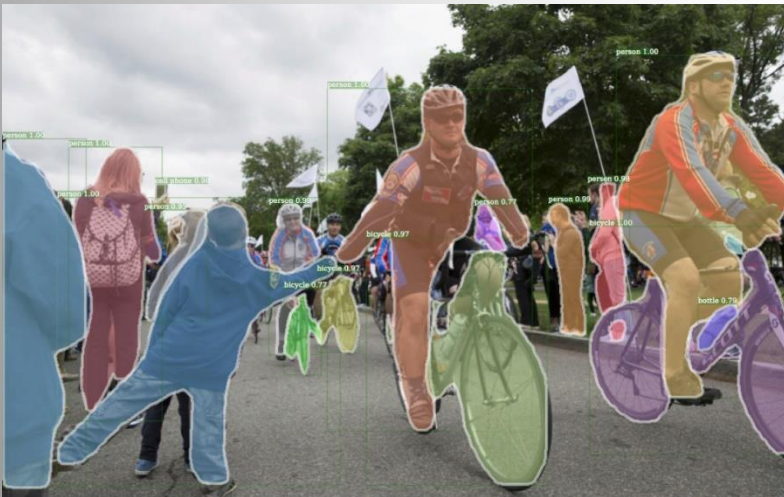
- Médical: <http://www.healthcare.siemens.fr/>
- Capture de Mouvement:
<http://www.4dviews.com/>
- Reconstruction 3D
 - [Microsoft Kinect](#) (structured light)
 - <http://www.ign.fr/>
- Vidéosurveillance <http://www.evitech.com>
- OCR (poste, [plaques immatriculations](#))
- Contrôle et mesures ([R&D vision](#))

Autres exemples

- Effets spéciaux:
 - Image stabilisation, correction
 - Object/background removal, replacement
 - Artificial makeup, motion capture
 - 3d camera solver and object insertion
 - [Mocha](#), [After effect](#), [Nuke](#), [Natron](#)
- Advanced image processing
 - Photoshop, gimp, paint.net

Deep Learning

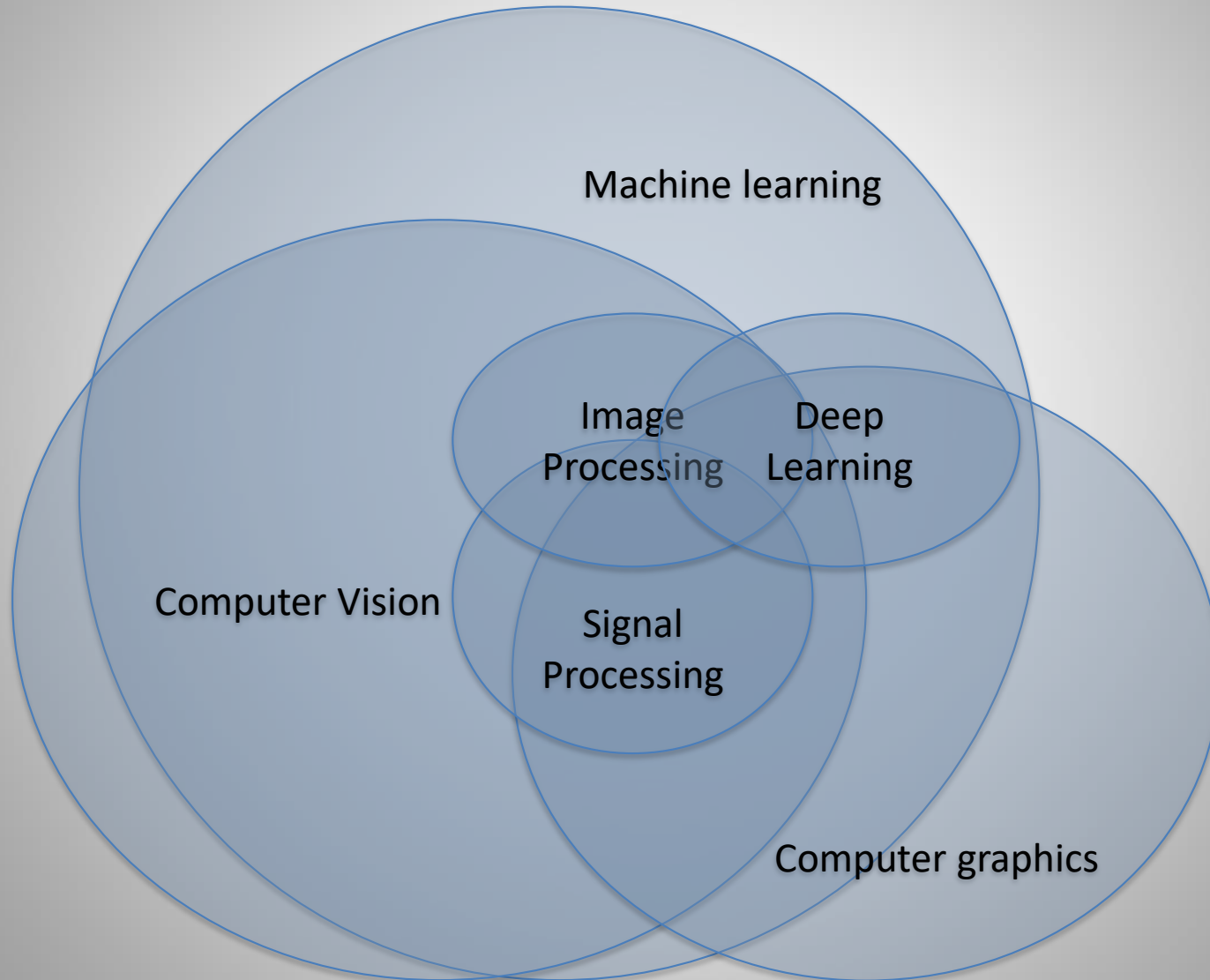
- Libraries: TensorFlow, Caffe, Torch, Detectron
- Yolo: <https://pjreddie.com/darknet/yolo/>
- OpenPose: <https://github.com/CMU-Perceptual-Computing-Lab/openpose>



Domaines Connexes

Domain	Input	Output
Image processing	Image	Image
Signal processing	Signal	Signal, quantitative information, e.g. Peak location,
Computer vision	Image/video	Image, quantitative/qualitative information, e.g. size, color, shape, classification, etc...
Machine learning	Any feature signal, from e.g. image, video, sound, etc..	Signal, quantitative/qualitative information, image,...
Computer graphics	3D models, textures, lightings, data..	Image, video, stereoscopic, 360°, video games

Domaines Connexes



Quelques entreprises 06

- **Bentley (Ex: Acute3D):** reconstruction 3D
- **Digital Barriers (Ex Kineo):** video surveillance
- **Airbus, Thales:** Imagerie satellite
- **Median Technologies:** médical
- **Therapixel:** médical
- **Optis:** simulation lumière
- **Lm3labs:** interfaces interactives
- **Robocortex:** SDK

Introduction à la vision par ordinateur

- Book: [Richard Szeliski](#), [Scott Krig](#)
- Quelques cours:
 - [Fei fei Li](#) - Stanford
 - [James Hays](#) Georgia Tech
 - [Marc Pollefeys](#) ETH
 - [Derek Hoiem](#) Urbana-Champaign
- Algèbre linéaire
 - [Stanford review](#)
 - [Matrix cookbook](#)

Comprendre l'image

Hauteur de cette femme?

Hauteur de la camera?

Rotation de la
caméra?

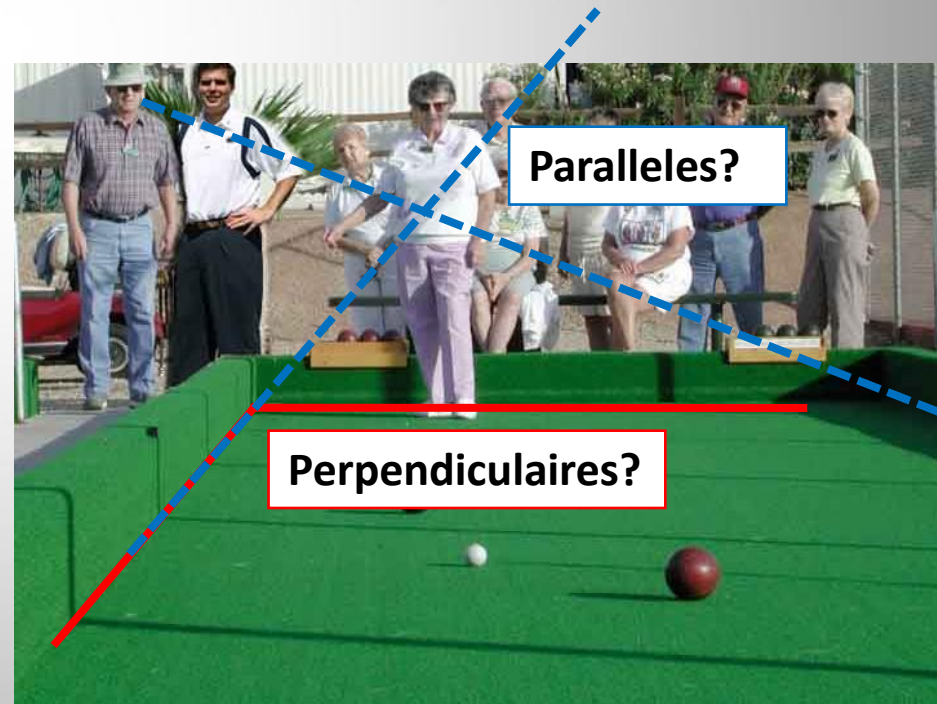
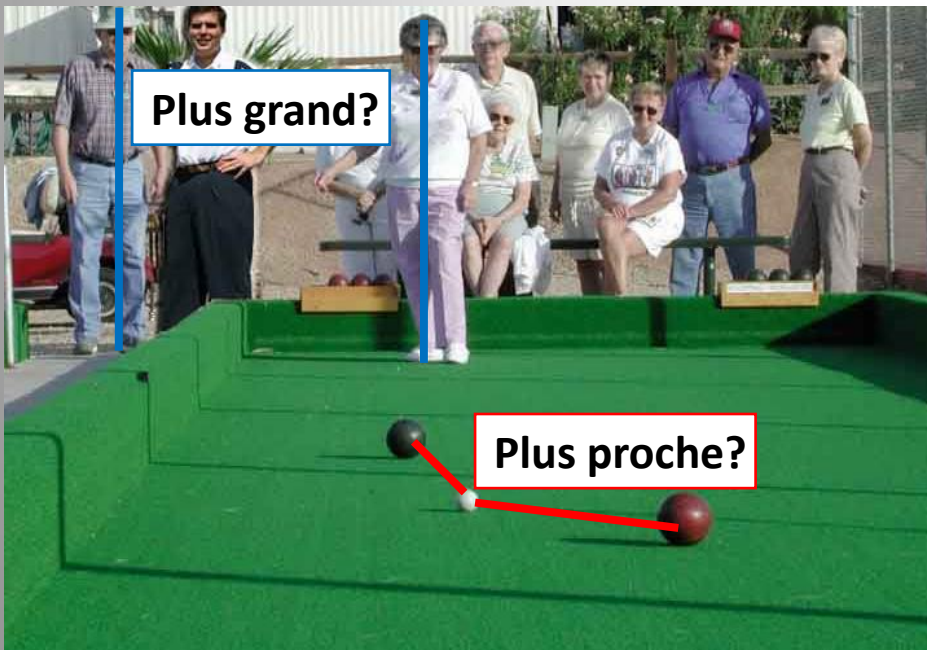
Focale de la camera?

Quelle balle est la plus proche?



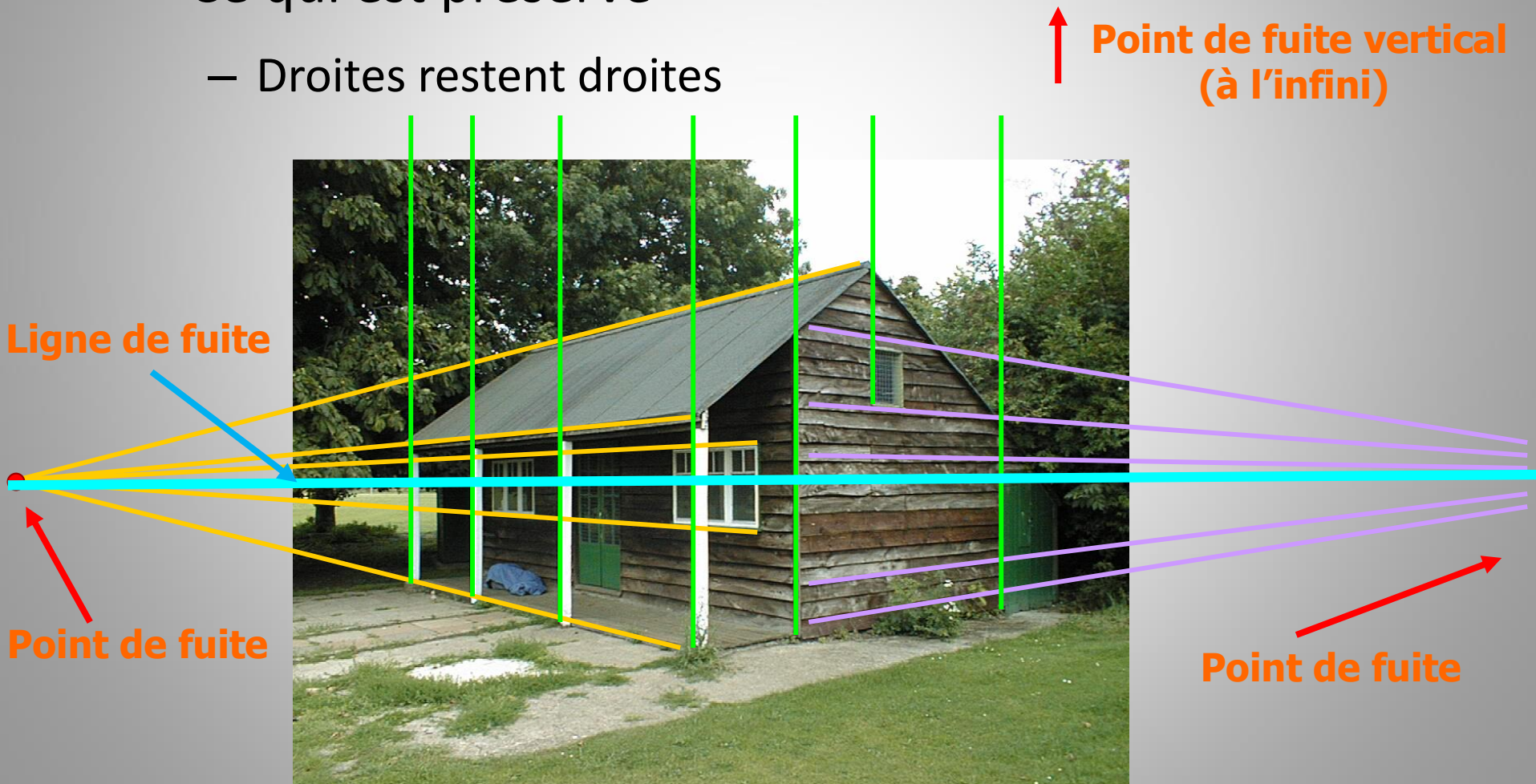
Comprendre l'image

- Ce qui est perdu
 - Longueurs
 - Angles



Comprendre l'image

- Ce qui est préservé
 - Droites restent droites



Comprendre l'image

- Propriété géométriques de la projection
 - Points restent des points
 - Droites restent droites
 - Plans donnent l'image complète
ou demi-plan
 - Polygones donnent des polygones
- Cas dégénérés:
 - Droite à travers centre optique donne un point
 - Plan à travers centre optique donne une droite

3D->image = projection

- Modèle de projection simple

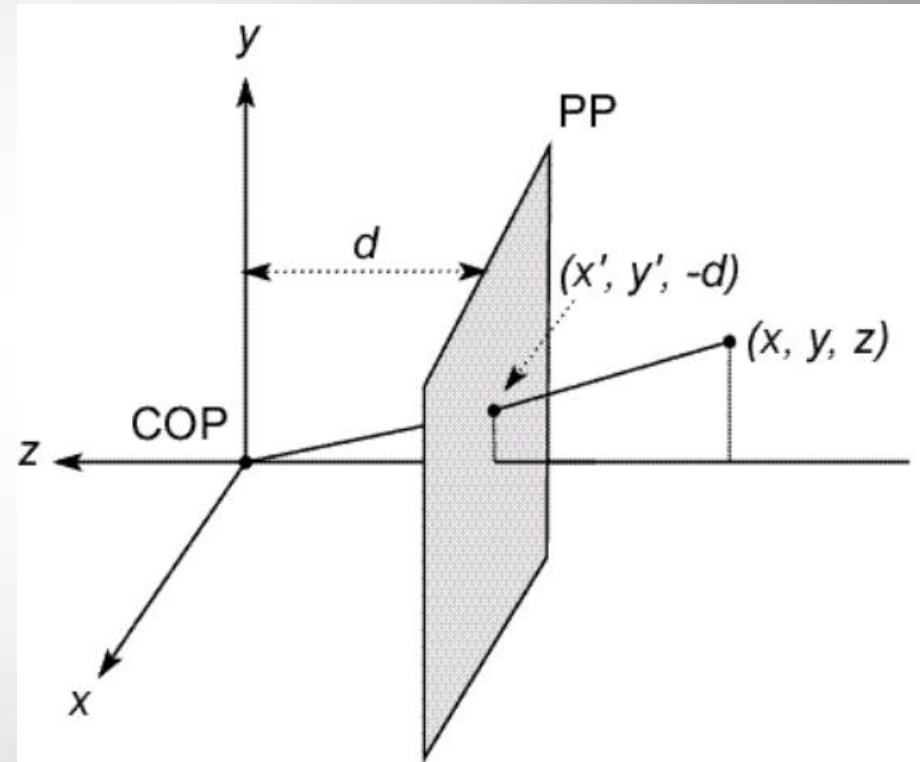
Équations projection:

$$(X, Y, Z) \rightarrow \left(-d \frac{X}{Z}, -d \frac{Y}{Z}, -d\right)$$

- On obtient:

$$(x', y') = \left(-d \frac{X}{Z}, -d \frac{Y}{Z}\right)$$

- Transformation non linéaire car division par Z
- Trick: coordonnées homogènes**



Coordonnées homogènes

- Conversion

Conversion en coordonnées homogènes

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Coordonnées homogènes
image

$$(x, y, z) \Rightarrow \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Coordonnées homogènes
scène

Conversion à partir des coordonnées homogènes

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow (x/w, y/w)$$

$$\begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \Rightarrow (x/w, y/w, z/w)$$

Coordonnées homogènes

- Line equation: $ax + by + c = 0$

$$line_i = \begin{bmatrix} a_i \\ b_i \\ c_i \end{bmatrix}$$

- Append 1 to pixel coordinate to get homogeneous coordinate

$$p_i = \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}$$

- Line given by cross product of two points

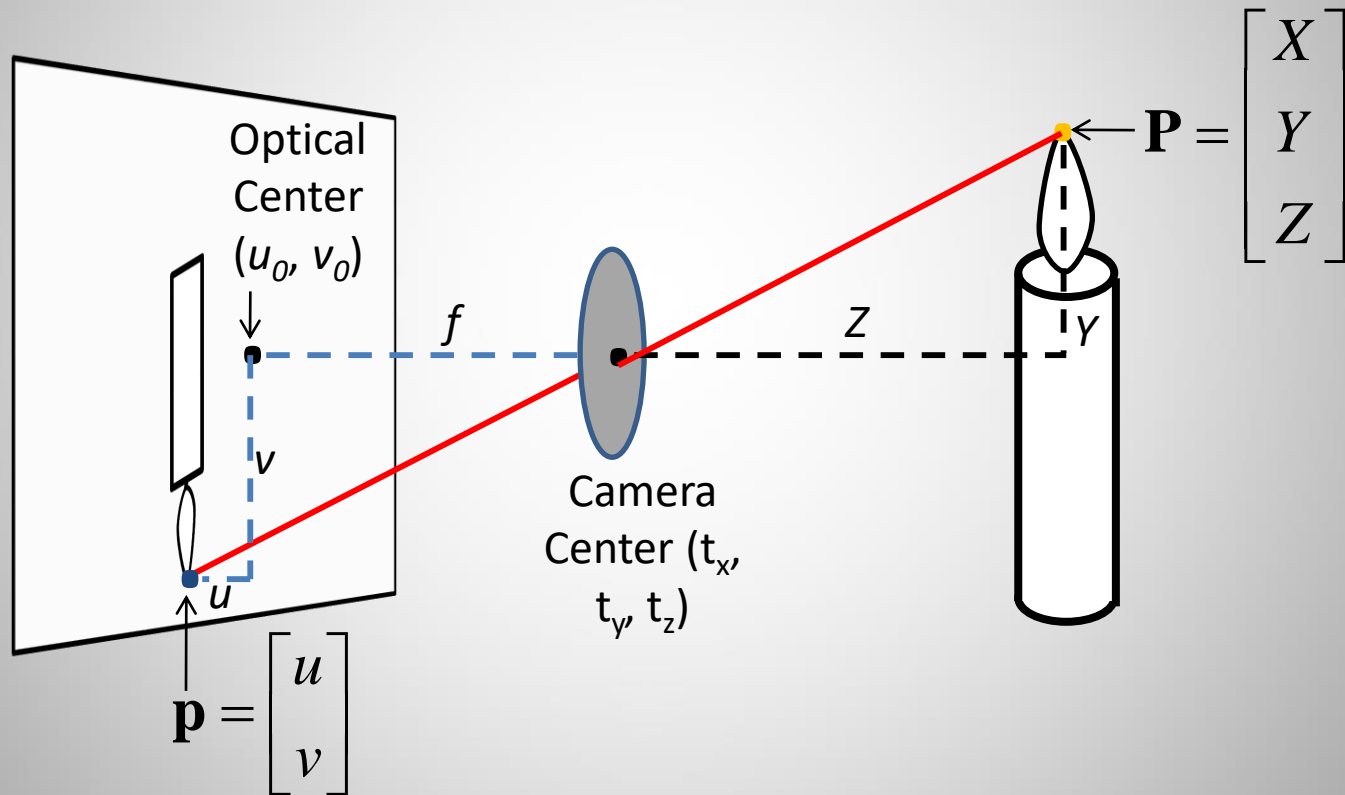
$$line_{ij} = p_i \times p_j$$

- Intersection of two lines given by cross product of the lines

$$q_{ij} = line_i \times line_j$$

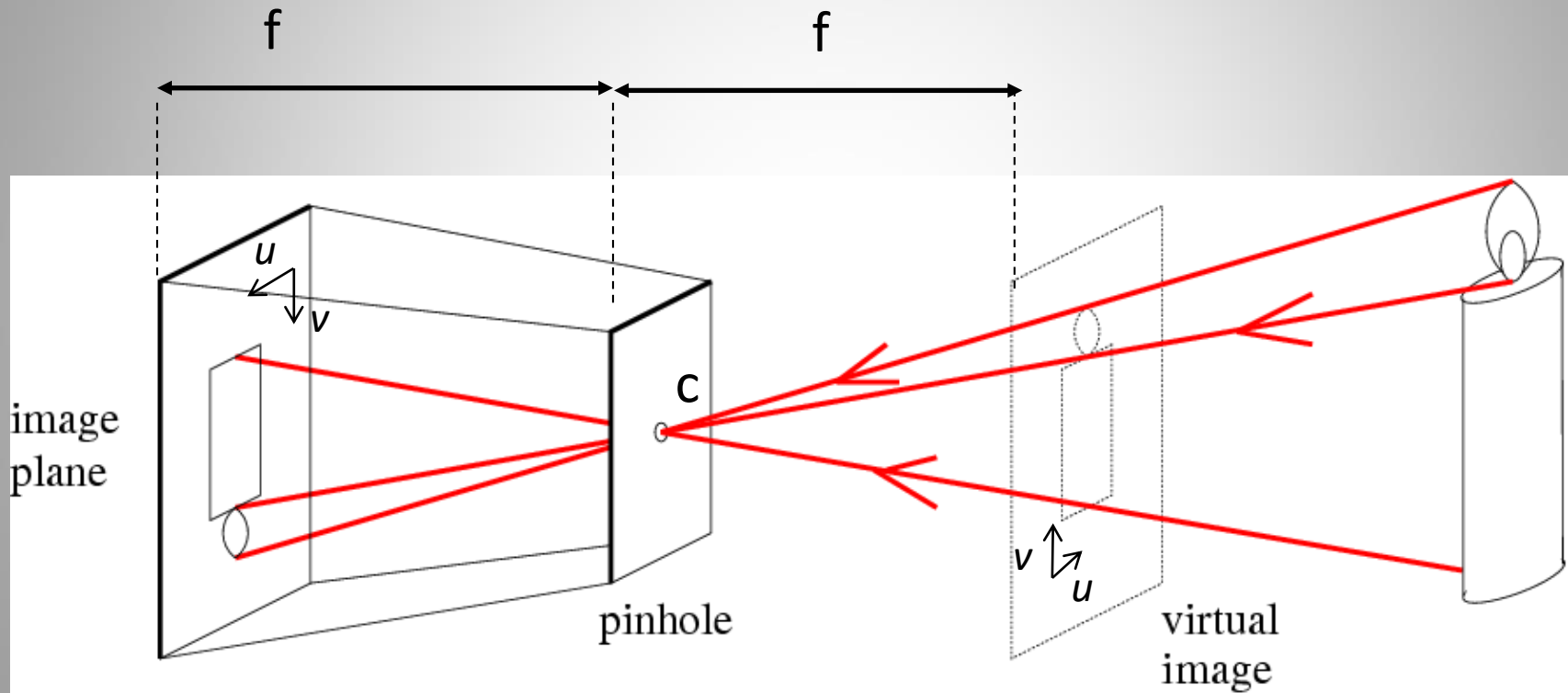
Formation de l'image

- Projection: world coordinates -> image coordinates



Formation de l'image

- Pinhole camera et image virtuelle

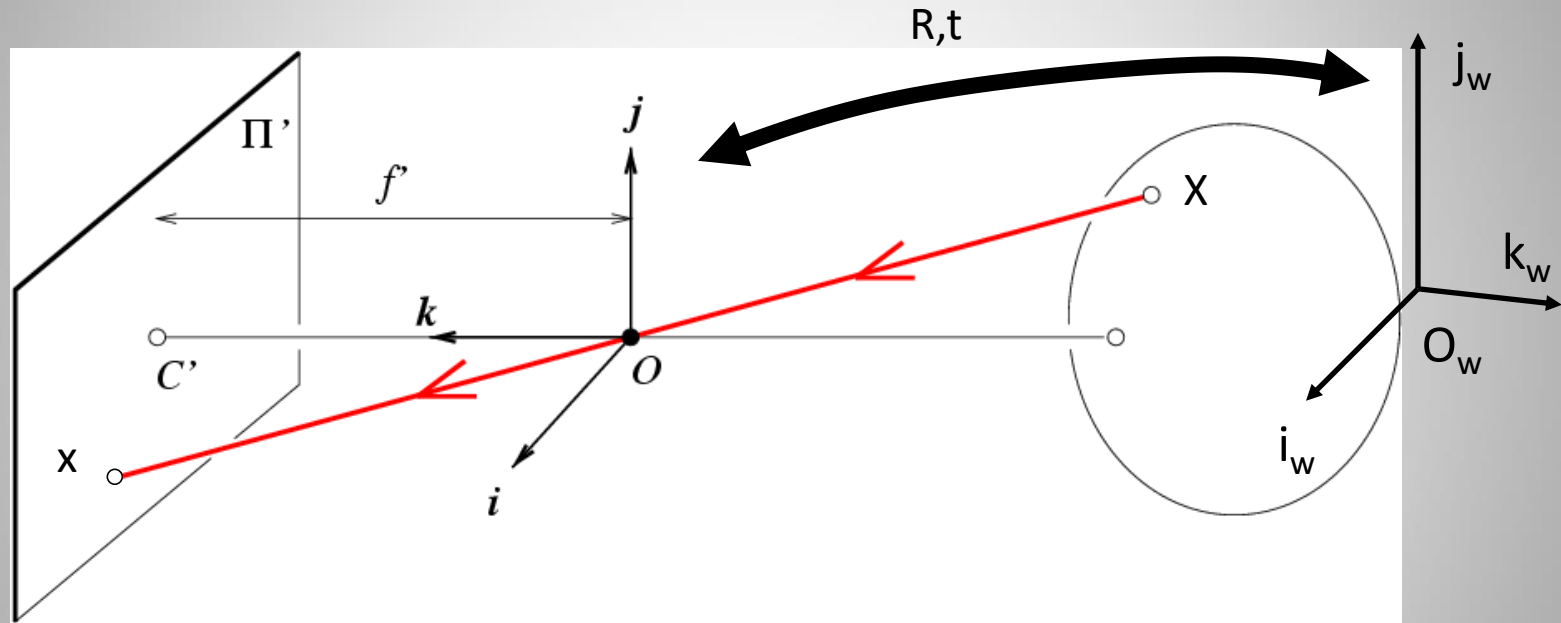


f = focal length

c = center of the camera

Formation de l'image

- Matrice de projection



$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}$$

\mathbf{x} : Image Coordinates: $(u, v, 1)$

\mathbf{K} : Intrinsic Matrix (3×3)

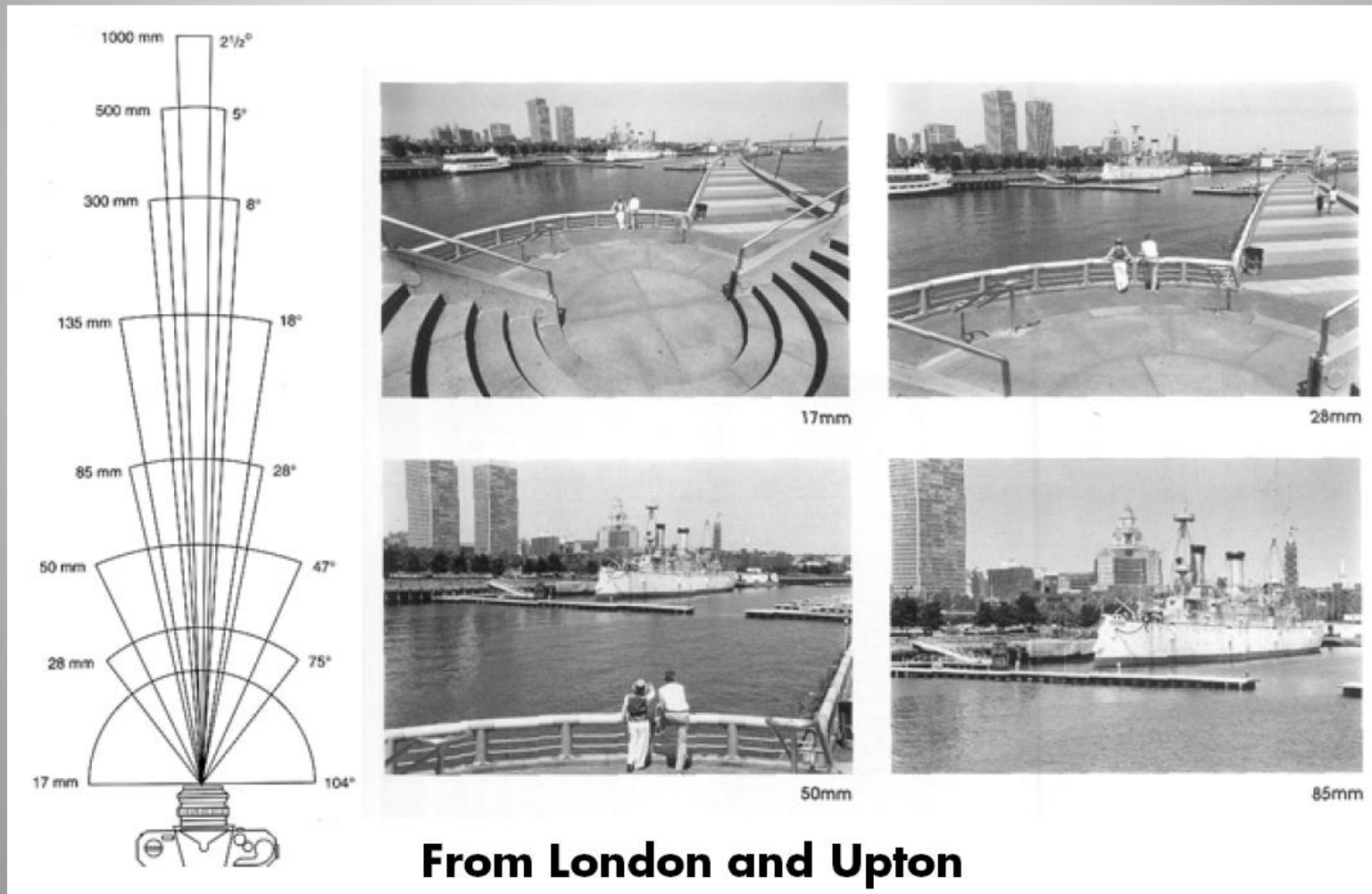
\mathbf{R} : Rotation (3×3)

\mathbf{t} : Translation (3×1)

\mathbf{X} : World Coordinates: $(X, Y, Z, 1)$

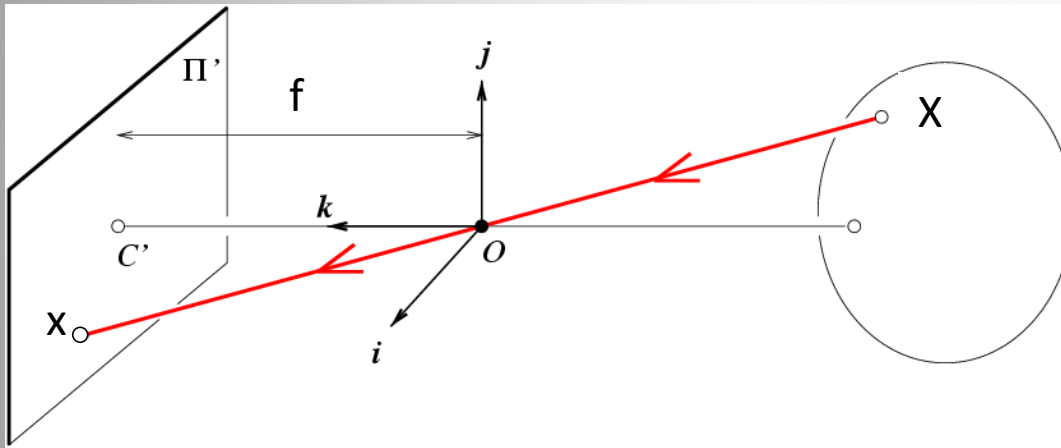
Formation de l'image

- Champs de vision (Zoom et focale f)



Formation de l'image

- Modèle simplifié avec seulement f



Hypothèses intrinsèques

- Aspect ratio de 1
- Centre optique en $(0,0)$
- Pixels carrés

Hypothèse extrinsèques

- Pas de rotation
- Camera en $(0,0,0)$

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{X} \Rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Formation de l'image

- En enlevant les hypothèses intrinsèques:
 - Centre optique non centré -> u_0, v_0
 - Pixels non carrés
 - Ajout de skew

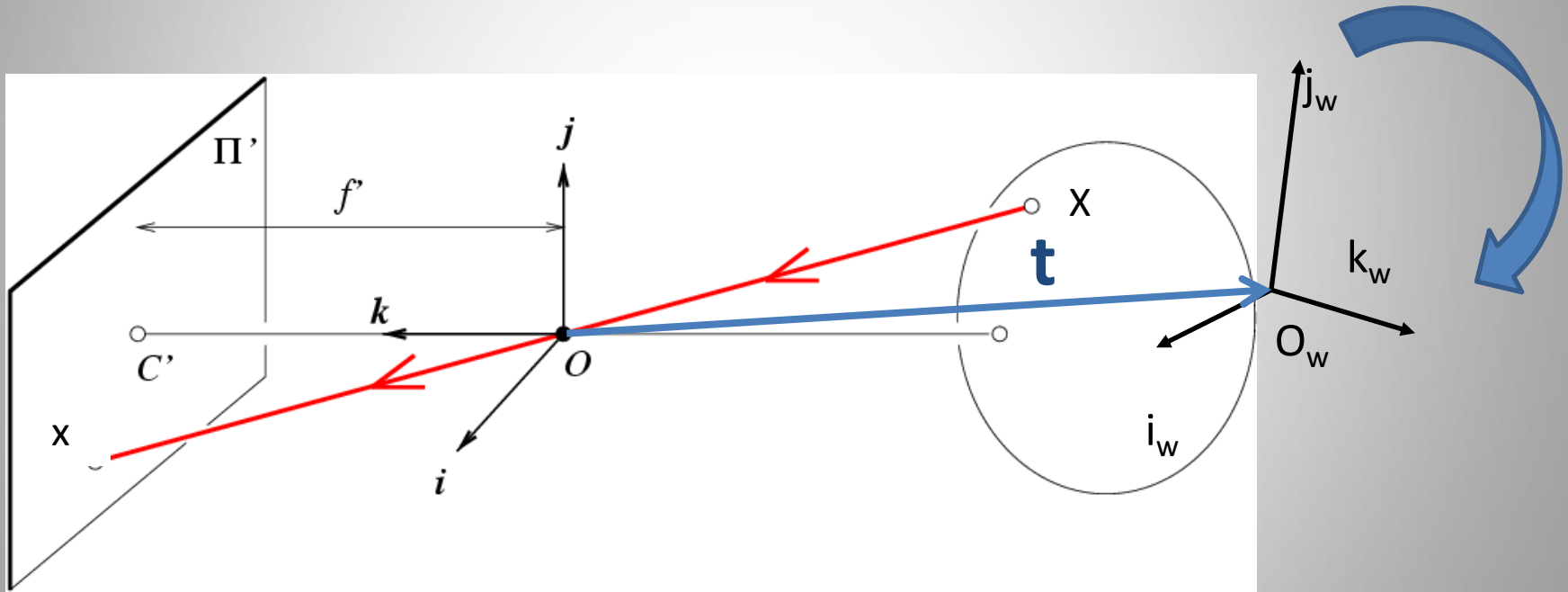
Hypothèse extrinsèques

- Pas de rotation
- Camera en $(0,0,0)$

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{X} \Rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 & 0 \\ 0 & \beta & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Formation de l'image

- Caméra orientée et translatée



Formation de l'image

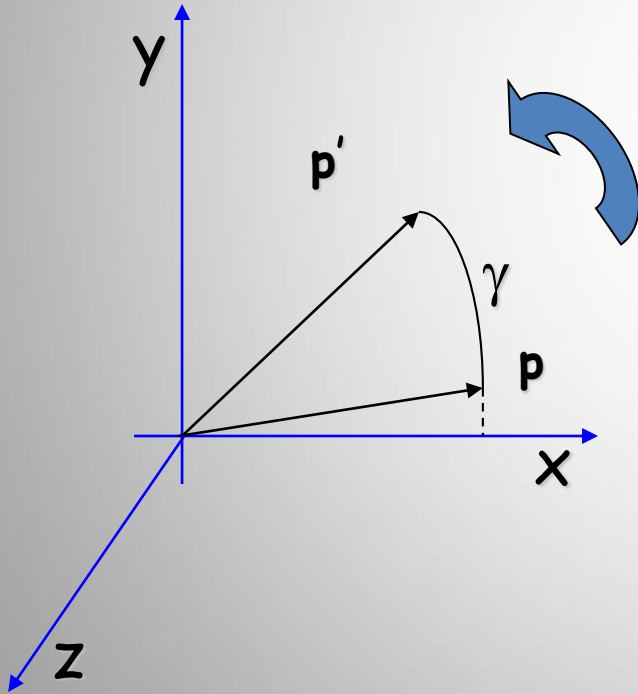
- On enlève l'hypothèse de Camera en (0,0,0)

Translation (tx,ty,tz)

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{t} \end{bmatrix} \mathbf{X} \Rightarrow \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & 0 & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Formation de l'image

- Rotation autour des axes, **counter-clockwise**:



$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha \\ 0 & \sin \alpha & \cos \alpha \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos \beta & 0 & \sin \beta \\ 0 & 1 & 0 \\ -\sin \beta & 0 & \cos \beta \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos \gamma & -\sin \gamma & 0 \\ \sin \gamma & \cos \gamma & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Formation de l'image

- Avec la rotation

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}$$

5



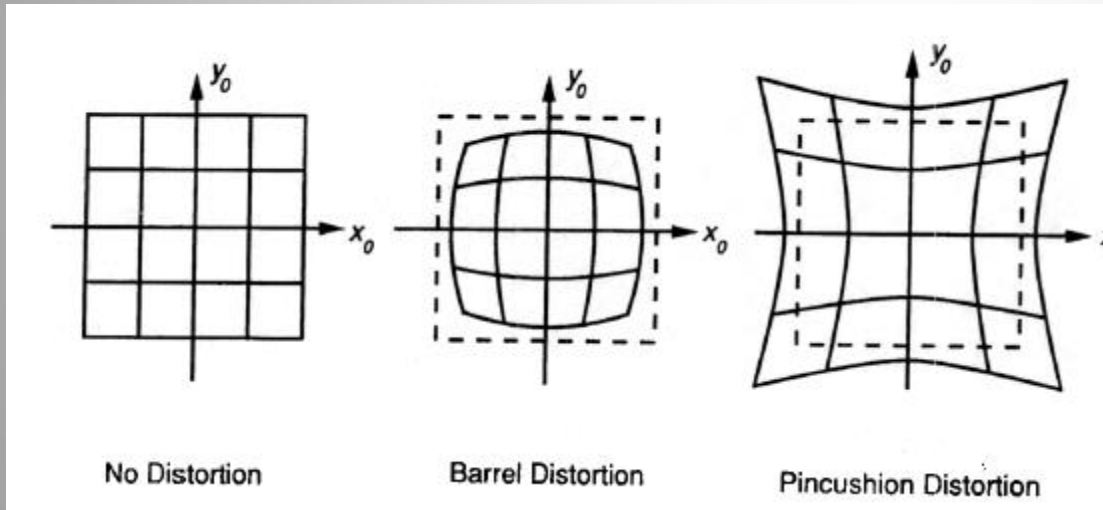
6

Degrès de liberté

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha & s & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Formation de l'image

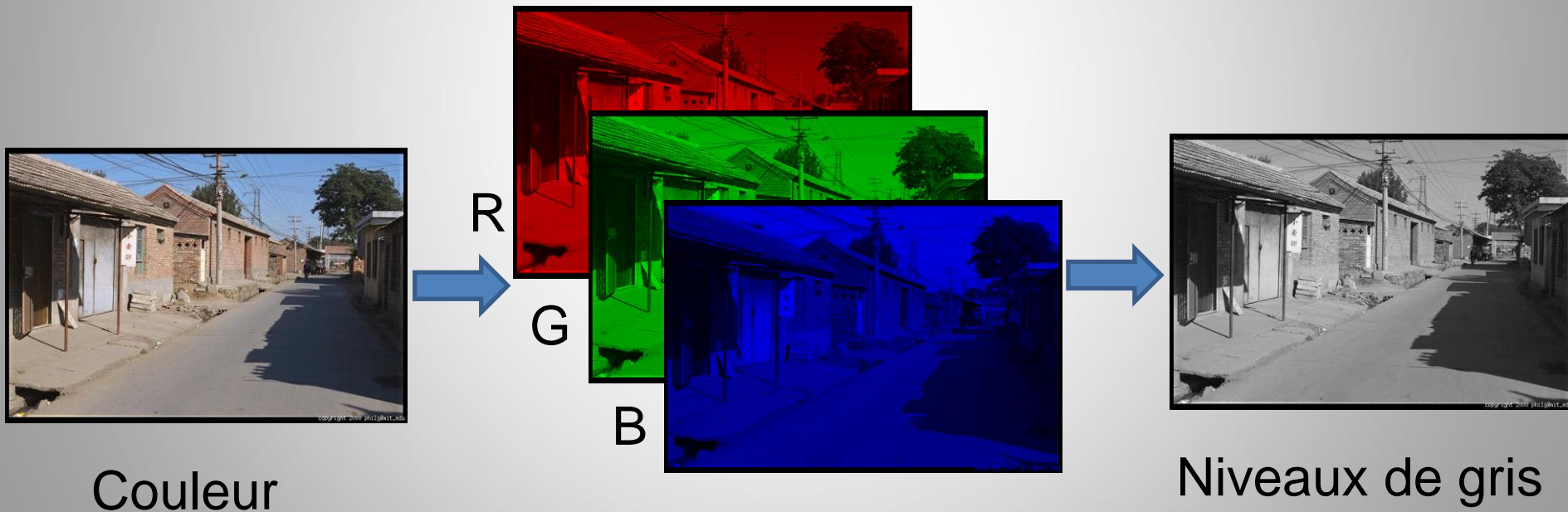
- Lentilles et distorsion



Corrected Barrel Distortion

Traitement de l'image

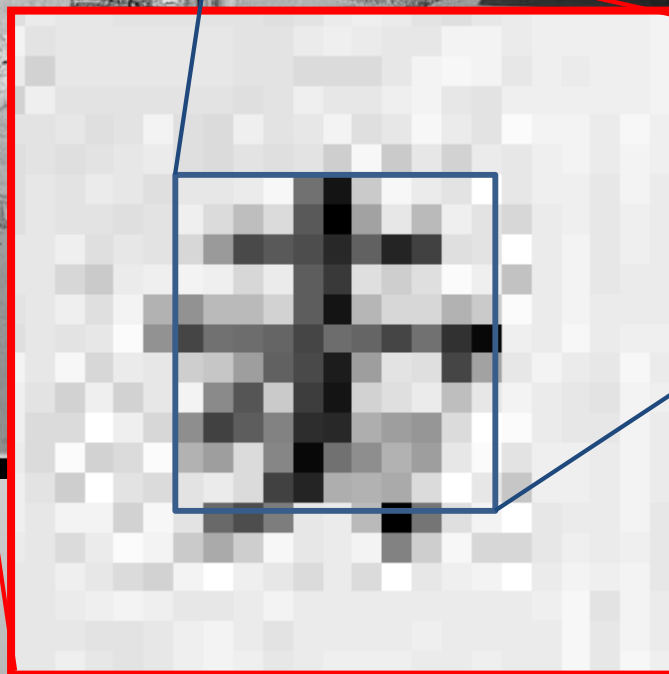
- Image de couleur = 3 images
- Algorithmes avec 1 entrée => Image de gris



Traitement de l'image



0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

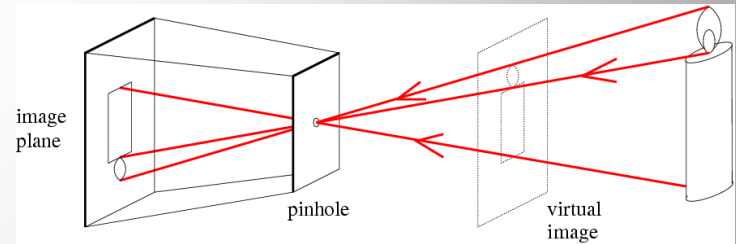
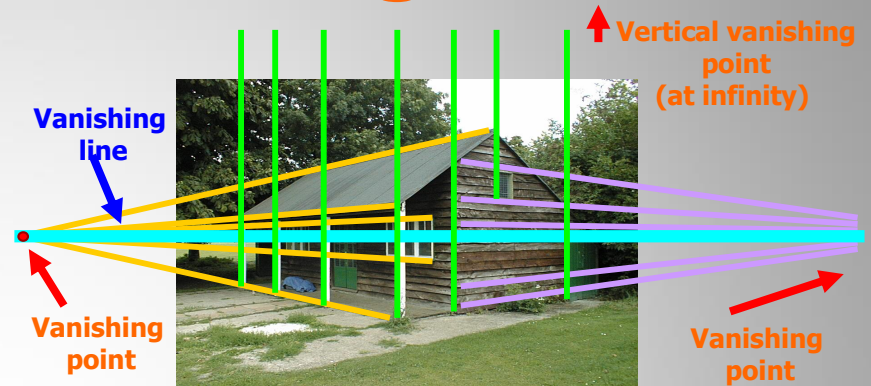


000 philg@mit.edu

Formation de l'image

Rappel

- Points et droites de fuite
- Modèle de caméra Pinhole et matrice de projection
- Coordonnées homogènes



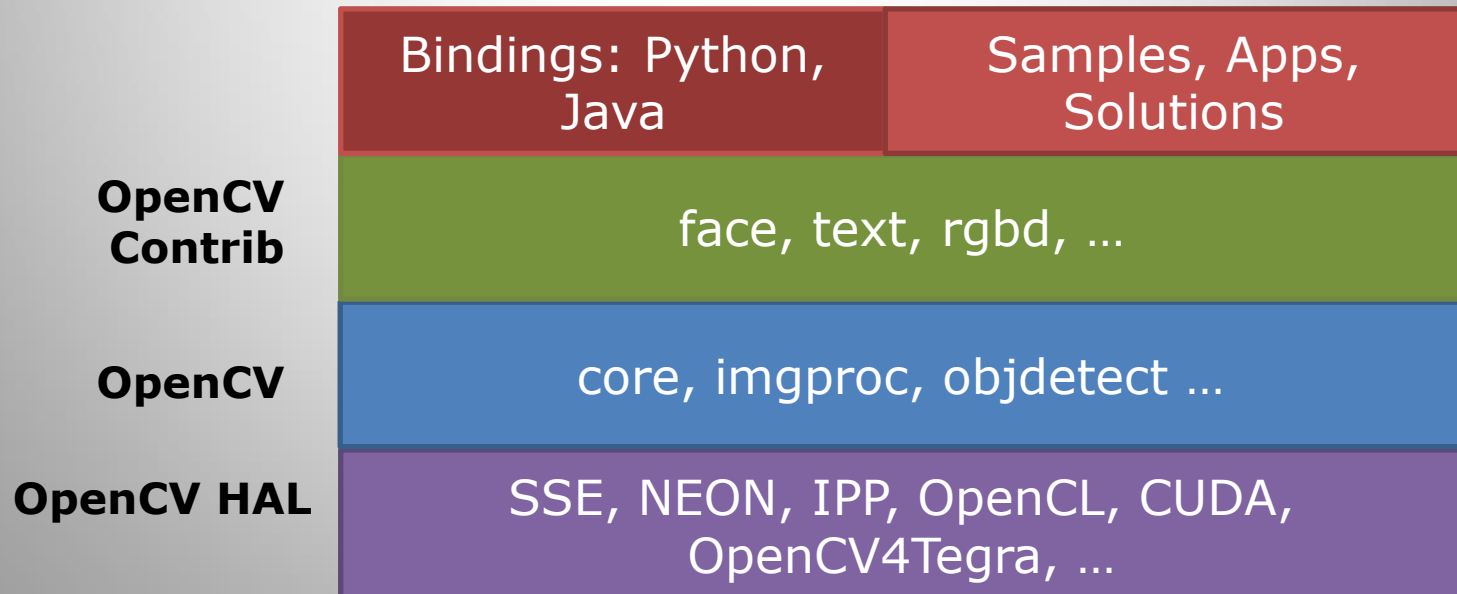
$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix} \mathbf{X}$$

$$(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



OpenCV

- <http://opencv.org/> et <https://github.com/Itseez/opencv>
- Librairie opensource C++
- BSD license, **10M** downloads, **500K+** lines of code
- Platforms:     **iOS**

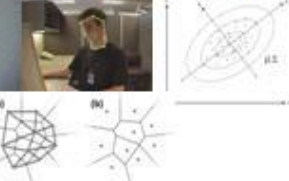
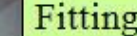
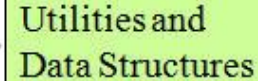
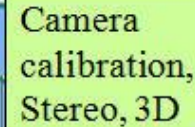
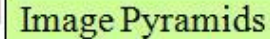
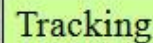
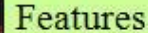
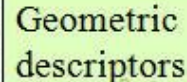
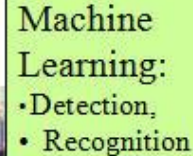
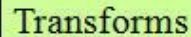
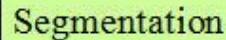
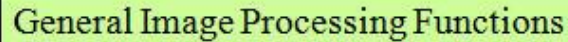




Robot support

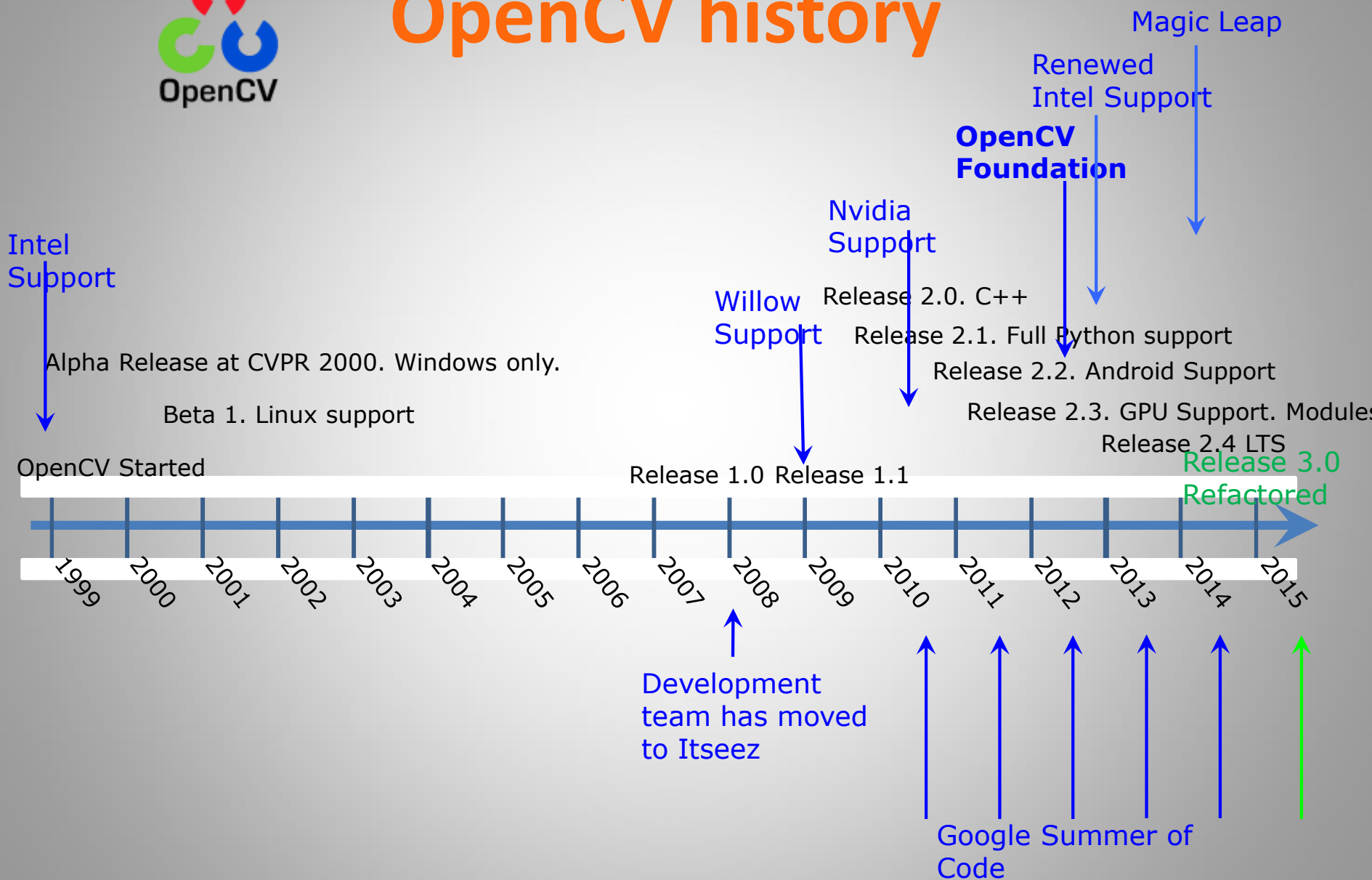


Robot support



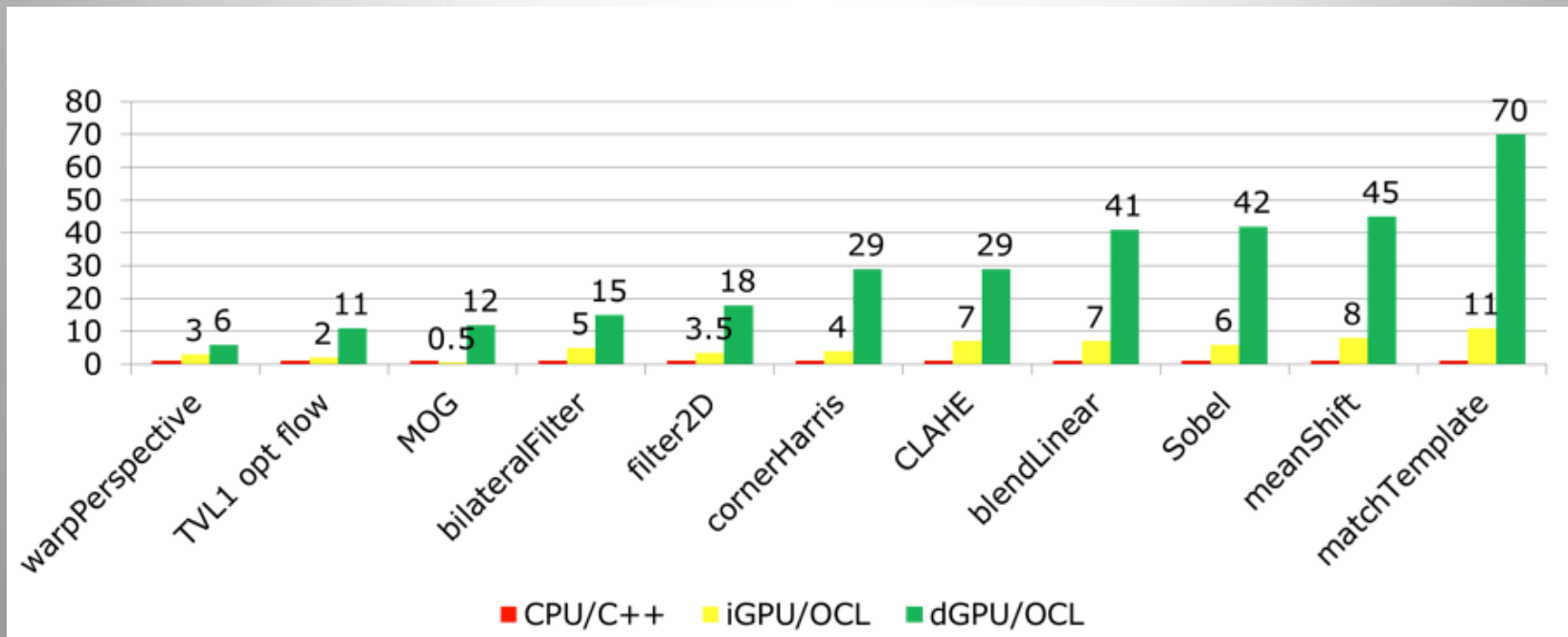


OpenCV history



Transparent API (T-API) for GPU acceleration

- single API entry for each function/algorithm – no specialized `cv::Canny`, `ocl::Canny`, `gpu::Canny` etc.
- uses dynamically loaded OpenCL runtime if available; otherwise falls back to CPU code. *Dispatching is at runtime, no recompilation needed!*
- ~100 functions are optimized



Coding in OpenCV

- OpenCV SheatCheet (attention version 2.4)

http://docs.opencv.org/3.0-last-rst/opencv_cheatsheet.pdf

-
- Type de données dans les images (Matrices)
Identificateur: **CV_<bit-dpth>{U|S|F}C(<nm_chnls>)**
 - Uchar: **CV_8UC1**
 - 3-elements float (RGB): **CV_32FC3**

Coding in OpenCV

- Création matrices/images

```
1 Mat mtx(3, 3, CV_32F); // make a 3x3 floating-point matrix
2 Mat cmtx(10, 1, CV_64FC2); // make a 10x1 2-channel floating-point
3                               // matrix (10-element complex vector)
4 Mat img(Size(1920, 1080), CV_8UC3); // make a 3-channel (color) image
5                               // of 1920 columns and 1080 rows.
6 Mat grayscale(image.size(), CV_MAKETYPE(image.depth(), 1)); // make a 1-channel image of
7                                                                // the same size and same
8                                                                // channel type as img
```

- Accès aux données

Access matrix elements

```
A33.at<float>(i,j) = A33.at<float>(j,i)+1;
```

```
Mat_<Vec3b>::iterator it = image.begin<Vec3b>(),
    itEnd = image.end<Vec3b>();
for(; it != itEnd; ++it)
    (*it)[1] ^= 255;
```

```
for(int y = 1; y < image.rows-1; y++) {
    Vec3b* prevRow = image.ptr<Vec3b>(y-1);
    Vec3b* nextRow = image.ptr<Vec3b>(y+1);
    for(int x = 0; x < image.cols; x++)
        for(int c = 0; c < 3; c++)
            dyImage.at<Vec3b>(y,x)[c] =
                saturate_cast<uchar>(
                    nextRow[x][c] - prevRow[x][c]);
}
```

Coding in OpenCV

- OpenCL-aware code OpenCV-2.x

```
// initialization
VideoCapture vcap(...);
ocl::OclCascadeClassifier fd("haar_ff.xml");
ocl::oclMat frame, frameGray;
Mat frameCpu;
vector<Rect> faces;
for(;;){
    // processing loop
    vcap >> frameCpu;
    frame = frameCpu;
    ocl::cvtColor(frame, frameGray, BGR2GRAY);
    ocl::equalizeHist(frameGray, frameGray);
    fd.detectMultiScale(frameGray, faces, ...);
    // draw rectangles ...
    // show image ...
}
```

- OpenCL-aware code OpenCV-3.x

```
// initialization
VideoCapture vcap(...);
CascadeClassifier fd("haar_ff.xml");
UMat frame, frameGray; // the only change from plain CPU version
vector<Rect> faces;
for(;;){
    // processing loop
    vcap >> frame;
    cvtColor(frame, frameGray, BGR2GRAY);
    equalizeHist(frameGray, frameGray);
    fd.detectMultiScale(frameGray, faces, ...);
    // draw rectangles ...
    // show image ...
}
```

OpenCV informations

- <http://opencv.org/>
- <http://docs.opencv.org/>
- <https://github.com/opencv/opencv/wiki>
- <https://www.intorobotics.com/opencv-tutorials-best-of/>
- Gsoc 2020 (DeadLine 16/31 mars):
https://github.com/opencv/opencv/wiki/GSoC_2020
- <https://www.youtube.com/watch?v=OUbUFn71S4s>

Compilation d'OpenCV

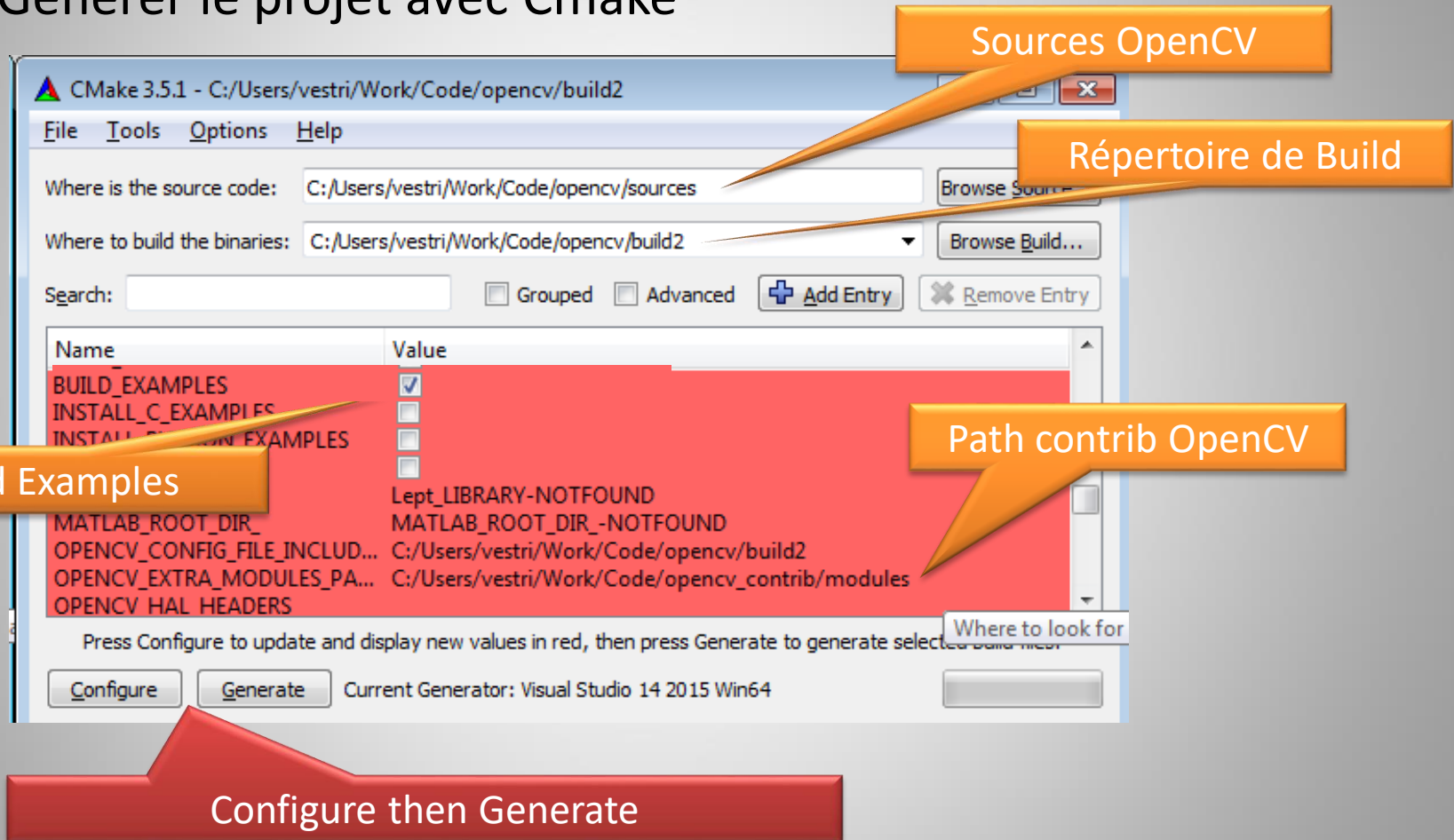
- Récupérez OpenCV, (contribs) et CmakeGUI
 - <http://opencv.org/>
 - https://github.com/Itseez/opencv_contrib
 - <https://cmake.org/>
- Mettre OpenCV (et OpenCVContrib) dans un répertoire de travail
- Créer répertoire build2 dans répertoire OpenCV

- openCV
 - build
 - build2
 - Sources
- Opencv_contribs

- Installer Cmake

Compilation d'OpenCV

- Générer le projet avec Cmake



- Ouvrir OpenCV.sln et compiler en release

Exercice dernier cours

- Déplacer et Animer un objet
- Utiliser Ground Plane Detection (Vuforia) et ajouter objet
- Ajouter un Canvas + image + fixer orientation smartphone
- Bouton pour lancer missile
- Système de particule pour fumée

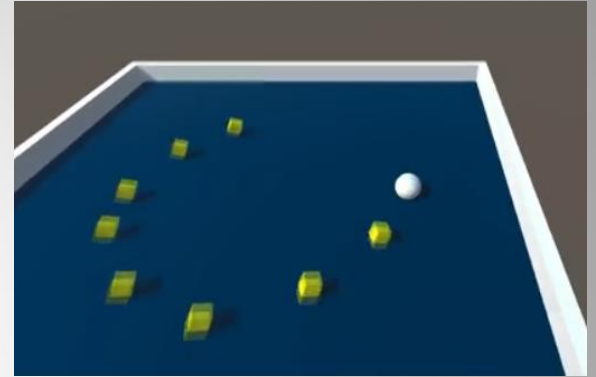


Matériel: <https://github.com/vestri/CoursAR>

Nouvel exercice

Roll a Ball

- Déplacer et Animer un objet
- Déplacer caméra avec objet
- Détecter collision d'objets
- Contrôle par smartphone ([Gyroscope.attitude](https://gyroscope.attitude))
- Score et construction du jeu



<https://learn.unity.com/project/roll-a-ball-tutorial>

Pour la semaine prochaine

Préparer un projet/demo

- Avec de la réalité augmentée, animation, effets...
- Unity&vuforia ou JS ou ce que vous voulez
- Si possible présenter sur mobile
- Expliquer en quelques mots votre projet, vos difficultés et comment résolues

Préparer quelques slides

- Projet et objectifs
- Développements, points techniques particulier, difficulté